Particle Filtering

Emin Orhan
eorhan@bcs.rochester.edu

August 11, 2012

Introduction: Particle filtering is a general Monte Carlo (sampling) method for performing inference in state-space models where the state of a system evolves in time and information about the state is obtained via noisy measurements made at each time step. In a general discrete-time state-space model, the state of a system evolves according to:

$$\mathbf{x}_k = \mathbf{f}_k(\mathbf{x}_{k-1}, \mathbf{v}_{k-1}) \tag{1}$$

where $\mathbf{x}_k$ is a vector representing the state of the system at time k, $\mathbf{v}_{k-1}$ is the state noise vector, $\mathbf{f}_k$ is a possibly non-linear and time-dependent function describing the evolution of the state vector. The state vector $\mathbf{x}_k$ is assumed to be latent or unobservable. Information about $\mathbf{x}_k$ is obtained only through noisy measurements of it, $\mathbf{z}_k$, which are governed by the equation:

$$\mathbf{z}_k = \mathbf{h}_k(\mathbf{x}_k, \mathbf{n}_k) \tag{2}$$

where $\mathbf{h}_k$ is a possibly non-linear and time-dependent function describing the measurement process and $\mathbf{n}_k$ is the measurement noise vector.

Optimal filtering: The filtering problem involves the estimation of the state vector at time k, given all the measurements up to and including time k, which we denote by $\mathbf{z}_{1:k}$. In a Bayesian setting, this problem can be formalized as the computation of the distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$, which can be done recursively in two steps. In the prediction step, $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ is computed from the filtering distribution $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ at time $k-1$:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})d\mathbf{x}_{k-1} \tag{3}$$

where $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$ is assumed to be known due to recursion and $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ is given by Equation 1. The distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k-1})$ can be thought of as a prior over $\mathbf{x}_k$ before receiving the most recent measurement $\mathbf{z}_k$. In the update step, this prior is updated with the new measurement $\mathbf{z}_k$ using the Bayes' rule to obtain the posterior over $\mathbf{x}_k$:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \propto p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \tag{4}$$

In general, the computations in the prediction and update steps (Equations 3-4) cannot be carried out analytically, hence the need for approximate methods such as Monte Carlo sampling. In some restricted cases, however, the computations in Equations 3-4 *can* be carried out analytically, as will be shown next.

The Kalman filter: If the filtering distribution at time $k-1$, $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$, is Gaussian, the filtering distribution at the next time step, $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ can be shown to be Gaussian as well if the following two conditions are satisfied: (1) the state evolution and measurement functions, $\mathbf{f}_k$ and $\mathbf{h}_k$, are linear; (2) the state noise and the measurement noise, $\mathbf{v}_k$ and $\mathbf{n}_k$, are Gaussian. In this case, the state and measurement

equations reduce to the following forms:

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + \mathbf{v}_{k-1} \quad (5)$$

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{n}_k \quad (6)$$

where $\mathbf{v}_{k-1}$ and $\mathbf{n}_k$ are Gaussian random variables and F_k and H_k are the state evolution and measurement matrices that are assumed to be known. The analytically tractable computation of the prediction and update equations in this linear Gaussian case leads to the well-known Kalman filter algorithm. It is a very useful and intellectually satisfying exercise to derive the prediction and update equations for the linear Gaussian system given in Equations 5-6.

Sequential importance sampling (SIS): When the prediction and update steps of the optimal filtering are not analytically tractable, one has to resort to approximate methods, such as Monte Carlo sampling. Sequential importance sampling (SIS) is the most basic Monte Carlo method used for this purpose. In deriving the SIS algorithm, it is useful to consider the full posterior distribution at time $k-1$, $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$, rather than the filtering distribution, $p(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})$, which is just the marginal of the full posterior distribution with respect to $\mathbf{x}_{k-1}$. The idea in SIS is to approximate the posterior distribution at time $k-1$, $p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1})$, with a weighted set of samples $\{\mathbf{x}^i_{0:k-1}, w^i_{k-1}\}^N_{i=1}$, also called particles, and recursively update these particles to obtain an approximation to the posterior distribution at the next time step: $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$.

SIS is based on importance sampling. In importance sampling, one approximates a target distribution $p(x)$, using samples drawn from a proposal distribution $q(x)$. Importance sampling is generally used when it is difficult to sample directly from the target distribution itself, but much easier to sample from the proposal distribution. To compensate for the discrepancy between the target and proposal distributions, one has to weight each sample x^i by $w_i \propto \pi(x^i)/q(x^i)$ where $\pi(x)$ is a function that is proportional to $p(x)$ (i.e. $p(x) \propto \pi(x)$) and that we know how to evaluate. Applied to our posterior distribution at time $k-1$, importance sampling yields:

$$p(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \approx \sum_{i=1}^{N} w^i_{k-1} \delta_{\mathbf{x}^i_{0:k-1}} \quad (7)$$

where $\delta_{\mathbf{x}^i_{0:k-1}}$ is a delta function centered at $\mathbf{x}^i_{0:k-1}$. The crucial idea in SIS is to update the particles $\mathbf{x}^i_{0:k-1}$ and their weights w^i_{k-1} such that they would approximate the posterior distribution at the next time step: $p(\mathbf{x}_{0:k}|\mathbf{z}_{1:k})$. To do this, we first assume that the proposal distribution at time k can be factorized as follows:

$$q(\mathbf{x}_{0:k}|\mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) q(\mathbf{x}_{0:k-1}|\mathbf{z}_{1:k-1}) \quad (8)$$

so that we can simply augment each particle $\mathbf{x}^i_{0:k-1}$ at time $k-1$ with a new state $\mathbf{x}^i_k$ at time k sampled from $q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k})$. To update the weights, w^i_{k-1}, we note that according importance sampling, the weights of the particles at time k should be as follows:

$$w^i_k \propto \frac{p(\mathbf{x}^i_{0:k}|\mathbf{z}_{1:k})}{q(\mathbf{x}^i_{0:k}|\mathbf{z}_{1:k})} \quad (9)$$

We will not give the full derivation here, but it turns out that it is possible to express Equation 9 recursively in terms of w^i_{k-1}. As usual, it is a very instructive exercise to derive this weight update equation by yourself. The final result turns out to be:

$$w^i_k \propto w^i_{k-1} \frac{p(\mathbf{z}_k|\mathbf{x}^i_k) p(\mathbf{x}^i_k|\mathbf{x}^i_{k-1})}{q(\mathbf{x}^i_k|\mathbf{x}^i_{0:k-1}, \mathbf{z}_{1:k})} \quad (10)$$

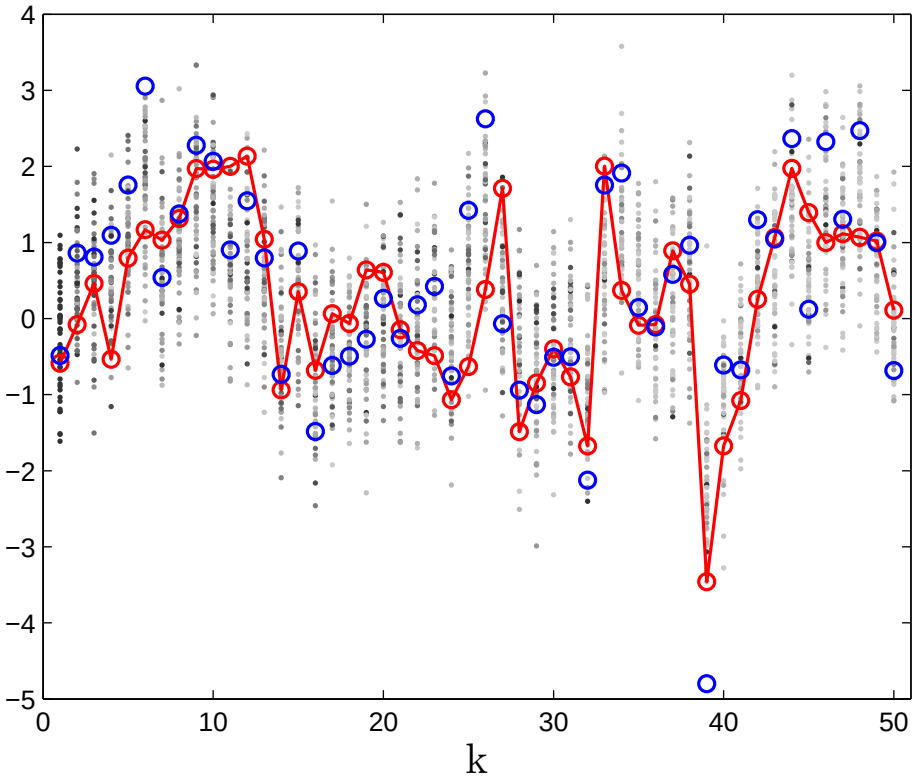

Figure 1: Demonstration of the SIS filtering algorithm on a linear Gaussian model. The red line shows the state, $\mathbf{x}_k$, of the system at each time step k; the blue circles show the measurements, $\mathbf{z}_k$, at the corresponding time steps. The gray dots show the particles generated by the SIS algorithm at each time step and the weights of the particles are represented by the gray levels of the dots, with darker colors corresponding to larger weights. In this example, we used $N = 50$ particles.

The weights $\{w_k^i\}_{i=1}^N$ are then normalized to sum to 1. If we also assume that $q(\mathbf{x}_k|\mathbf{x}_{0:k-1}, \mathbf{z}_{1:k}) = q(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{z}_k)$ so that the proposal at the next time step only depends on the most recent state and the most recent measurement, then we only need to store $\mathbf{x}_{k-1}^i$ and generate the particle at the next time step from the distribution $q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$. Thus, in this case, the update equations simplify to:

$$\mathbf{x}_k^i \sim q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k) \tag{11}$$

$$w_k^i \propto w_{k-1}^i \frac{p(\mathbf{z}_k|\mathbf{x}_k^i)p(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i)}{q(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k)} \tag{12}$$

Figure 1 shows the application of the SIS algorithm to a toy linear Gaussian model. Note that the weights of the particles, represented by the gray levels of the dots in this figure, are initially uniform, but as k increases, they gradually come to be dominated by a smaller and smaller number of particles. This illustrates the degeneracy problem, which we take up next.

Degeneracy problem: In practice, iteration of the update equations in 11-12 leads to a degeneracy problem where only a few of the particles will have a significant weight, and all the other particles will have very small weights. Degeneracy is typically measured by an estimate of the effective sample size:

$$N_{eff} = \frac{1}{\sum_{i=1}^N (w_k^i)^2} \tag{13}$$

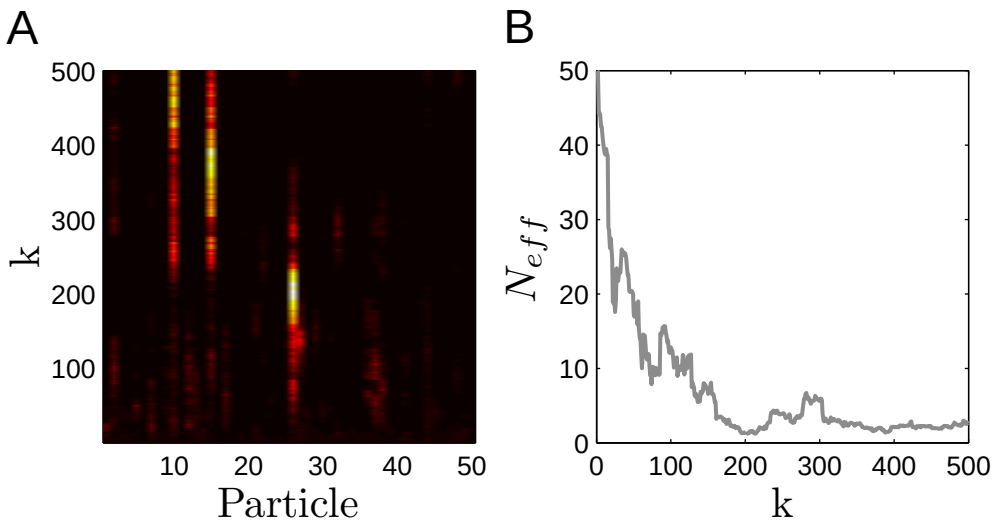

Figure 2: (A) The weights of all 50 particles (x-axis) at each time step k (y-axis). Hotter colors represent larger weights. (B) The effective sample size N_{eff} as a function of time step k.

where a smaller N_{eff} means a larger variance for the weights, hence more degeneracy.

Figure 2 illustrates the degeneracy problem in a toy example. In this example, an SIS filter with $N = 50$ particles was applied to a linear Gaussian state-space model for 500 time steps. Figure 2A shows the weights of all particles (x-axis) at each time step k (y-axis); Figure 2B plots the effective sample size N_{eff} as a function of time step k. As k increases, N_{eff} drops very quickly from an initial value of 50 to values smaller than 5, indicating that for large k, only a handful of particles have significant weights.

Resampling: One common way to deal with degeneracy is resampling. In resampling, one draws (with replacement) a new set of N particles from the discrete approximation to the filtering distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ (or to the full posterior distribution, if one is interested in computing the full posterior) provided by the weighted particles:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \delta_{\mathbf{x}_k^i} \tag{14}$$

Resampling is performed whenever the effective sample size N_{eff} drops below a certain threshold. Note that since resampling is done with replacement, a particle with a large weight is likely to be drawn multiple times and conversely particles with very small weights are not likely to be drawn at all. Also note that the weights of the new particles will all be equal to $1/N$. Thus, resampling effectively deals with the degeneracy problem by getting rid of the particles with very small weights. This, however, introduces a new problem, called the sample impoverishment problem. Because particles with large weights are likely to be drawn multiple times during resampling, whereas particles with small weights are not likely to be drawn at all, the diversity of the particles will tend to decrease after a resampling step. In the extreme case, all particles might "collapse" into a single particle. This will negatively impact the quality of the approximation in Equation 14, as one will be trying to represent a whole distribution with a fewer number of distinct samples (in the limit of a single particle, one will be representing the full distribution with a single point estimate). We will mention below one possible way to deal with this sample impoverishment problem, but we first briefly review a quite popular particle filtering algorithm that uses resampling.

Sampling importance resampling (SIR) algorithm: Most particle filtering algorithms are variants of the basic SIS algorithm considered above. The SIR algorithm can be seen as a variant of SIS where the proposal distribution $q(\mathbf{x}_k|\mathbf{x}_{k-1}^i, \mathbf{z}_k)$ is taken to be the state transition distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ and

resampling is applied at every iteration. Thus, in the SIR algorithm, the update equations for the particles (Equations 11-12 in the SIS algorithm) reduce to:

$$\mathbf{x}_k^i \sim p(\mathbf{x}_k|\mathbf{x}_{k-1}^i) \tag{15}$$
$$w_k^i \propto p(\mathbf{z}_k|\mathbf{x}_k^i) \tag{16}$$

These updates are then followed by a resampling step at each iteration. Note that the w_{k-1}^i term disappears in the weight update equation in 16, because after resampling at time $k-1$, all weights w_{k-1}^i become equal to $1/N$. The main advantage of the SIR algorithm is that it is extremely simple to implement, since it only requires sampling from the distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ and evaluating $p(\mathbf{z}_k|\mathbf{x}_k^i)$ (plus resampling at every iteration). Its disadvantages are (i) the independence of the proposal distribution $p(\mathbf{x}_k|\mathbf{x}_{k-1}^i)$ from the observations $\mathbf{z}_k$ (this means that the states are updated without directly taking into account the information provided by the observations); (ii) the sample impoverishment problem which can be severe due to resampling at every iteration.

Figure 3A shows the application of an SIR filter to a simple linear Gaussian system. To illustrate the effects of resampling, Figure 3B shows the results for an SIS filter applied to the same system. The SIS filter used the same proposal distribution as the SIR filter, but did not involve resampling. Note that in Figure 3B, the weight distribution is dominated by a single particle for later time steps, illustrating the degeneracy problem, whereas the particles in Figure 3A all have the same weight, $1/N$, due to resampling at each time step in the SIR algorithm. Although resampling deals with the degeneracy problem, it leads to the sample impoverishment problem, which can be observed in Figure 3 by noting that particle diversity is reduced in Figure 3A compared with Figure 3B.

We will now present another variant of the SIS algorithm, called the regularized particle filter, that also uses resampling, but unlike the SIR algorithm, effectively deals with the sample impoverishment problem introduced by resampling.

Regularized particle filter: The sample impoverishment problem during resampling arises because we try to approximate a continuous distribution with a discrete one (Equation 14): with a discrete distribution, there is a strictly positive probability that two samples drawn from the distribution will be identical, whereas with a continuous distribution, no two samples drawn from the distribution will be identical. In the regularized particle filter, one approximates the filtering distribution in Equation 14 (or the full posterior distribution) with a kernel density estimate using the particles rather than directly with the particles themselves. Thus, in resampling, Equation 14 is replaced by:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^i K(\mathbf{x}_k - \mathbf{x}_k^i) \tag{17}$$

where $K(\mathbf{x}_k - \mathbf{x}_k^i)$ is a kernel function centered at $\mathbf{x}_k^i$. For the case of equal weights, it turns out to be possible to analytically determine the shape and the bandwidth of the optimal kernel function that minimizes a certain distance measure between the true filtering distribution and the kernel density estimate obtained via the approximation in Equation 17. The optimal kernel function in this case is known as the Epanechnikov kernel (see Arulampalam, Maskell, Gordon & Clapp (2002) for details).

The complete regularized particle filter algorithm then consists of the state and weight update equations of the SIS filter (Equations 11-12), combined with resampling N new particles from Equation 17 whenever the effective sample size N_{eff} falls below a certain threshold N_T.

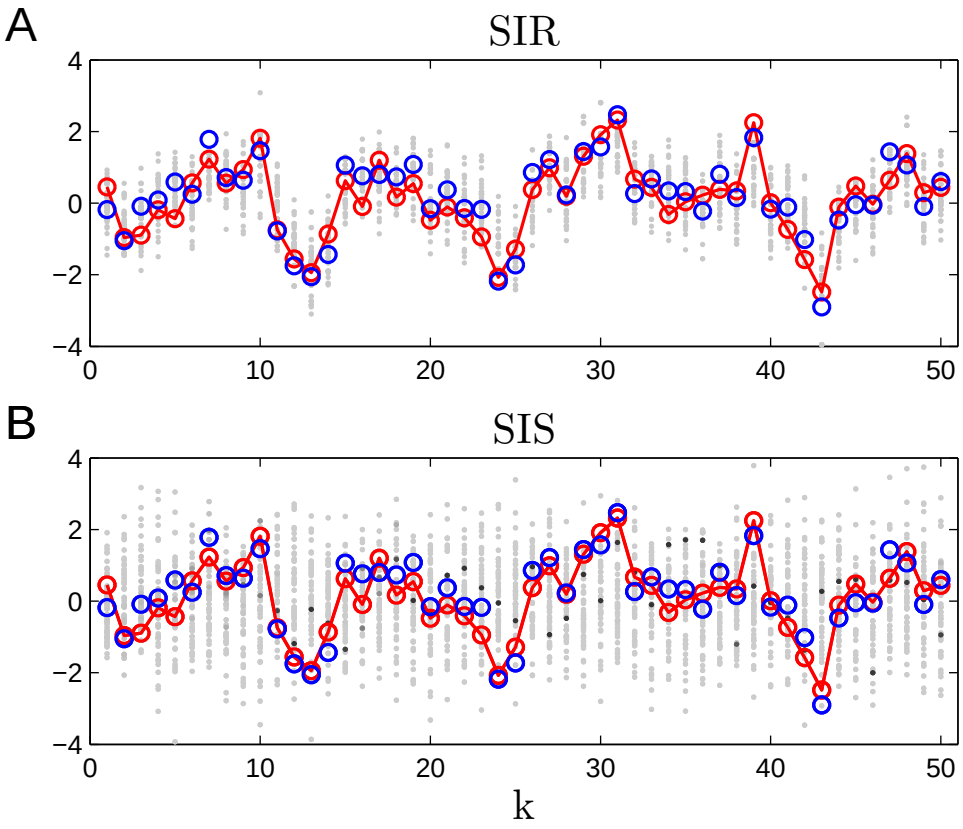

Figure 3: Comparison of SIR with SIS. (A) An SIR filter applied to a linear Gaussian system. The red line shows the state, $\mathbf{x}_k$, of the system at each time step k; the blue circles show the measurements, $\mathbf{z}_k$, at each time step. The gray dots show the particles generated by the SIR algorithm at each time step and the weights of the particles are represented by the gray levels of the dots, with darker colors corresponding to larger weights. (B) An SIS filter applied to the same system. The SIS filter used the same proposal distribution as the SIR filter, but did not involve resampling. In this example, we used $N = 50$ particles in both cases.

References

[1] Arulampalam, M.S., Maskell, S., Gordon, N. & Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*. 50(2):174-188.